APPENDIX A

```
/*
 * FUZZY CD ID
 * (c) 1996 ION
 *
 *
 * by Ty Roberts
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

struct fuzzyCDid {
        short                   numTracks;    // start time in milliseconds
        unsigned short                  fuzzlength[100];
};

typedef struct fuzzyCDid fuzzyCDid, *fuzzyCDidPtr;

// structure of a cd track with all times stored in milliseconds

struct cdtrack {
        long    beginMs;        // start time in milliseconds
        long    endMs;                  // end time in milliseconds
        long    lengthMs;       // length in milliseconds
};

typedef struct cdtrack cdtrack, *cdTrackPtr;

struct cd {
        short   numTracks;
        cdtrack         track[100];
};

typedef struct cd cd, *cdPtr;

void CreateFuzzyId( fuzzyCDidPtr fid, cdPtr cd  );

float   FuzzyMatch( fuzzyCDidPtr fid1, fuzzyCDidPtr fid2 );

// SUBROUTINES

void CreateFuzzyId( fuzzyCDidPtr fid, cdPtr cd  )
```

```
{
        long            i;

        // first copy in the number of tracks
        fid->numTracks = cd->numTracks;

        for(i=0;i<fid->numTracks;i++) {
                // shift left and create a MSB length thats not exact
                fid->fuzzlength[i] = (short)(cd->track[i].lengthMs>>8);
        }
}

float   FuzzyMatch( fuzzyCDidPtr fid1, fuzzyCDidPtr fid2 )
{
        long            fidmatcherr = 0,  fidmatchtotal = 0;
        short           i, trackcnt;
        float           matchpercent;

        // find the larger number of tracks
        trackcnt = fid1->numTracks<fid2->numTracks ? fid2->numTracks :
fid1->numTracks;

        // cycle thru the tracks accumulating error and total comparedtimes
        for(i=0;i<trackcnt;i++) {
                if ((i < fid1->numTracks) && (i < fid2->numTracks)) {
                        fidmatcherr += abs(fid1->fuzzlength[i] - fid2->fuzzlength[i]);
                        fidmatchtotal += fid1->fuzzlength[i];
                } else if (i >= fid2->numTracks) {
                        fidmatcherr += fid1->fuzzlength[i];
                        fidmatchtotal += fid1->fuzzlength[i];
                } else if (i >= fid1->numTracks) {
                        fidmatcherr += fid2->fuzzlength[i];
                        fidmatchtotal += fid2->fuzzlength[i];
                }
        }

        if (fidmatcherr > 0) {
                matchpercent = 100 - (((float)fidmatcherr/(float)fidmatchtotal) *100);
        } else {
                matchpercent = 100;
        }
        return matchpercent;
}
```

```
void main(void)
{
        short i;
        float   matchpercent;

        // create global structures for two complete cds with up to 100 tracks
        cd      cd2id;
        fuzzyCDid fidcd2id;
        cd      cdFromDB;

        fuzzyCDid fidcdFromDB;

        printf ("Test #1 will compare two CDs that are exactly the same\n\n");

        // put in some test values for the cd track lengths
        // since these are in ms, its basically 60000 = 1 minute

        cd2id.track[0].lengthMs = 121323;
        cd2id.track[1].lengthMs = 234565;
        cd2id.track[2].lengthMs = 566437;
        cd2id.track[3].lengthMs = 245120;
        cd2id.track[4].lengthMs = 20000;
        cd2id.track[5].lengthMs = 120386;
        cd2id.track[6].lengthMs = 323453;
        cd2id.numTracks = 7;

        for(i=1;i<cd2id.numTracks;i++) {
                printf ("CD #1: Track = %d   length in minutes = %f\n",
                        i, (float)cd2id.track[i].lengthMs/60000.0);
        }
        printf("\n");

        cdFromDB.track[0].lengthMs = 121323;
        cdFromDB.track[1].lengthMs = 234565;
        cdFromDB.track[2].lengthMs = 566437;
        cdFromDB.track[3].lengthMs = 245120;
        cdFromDB.track[4].lengthMs = 20000;
        cdFromDB.track[5].lengthMs = 120386;
        cdFromDB.track[6].lengthMs = 323453;
        cdFromDB.numTracks = 7;

        for(i=1;i<cdFromDB.numTracks;i++) {
                printf ("CD #2: Track = %d   length in minutes = %f\n",
                        i, (float)cdFromDB.track[i].lengthMs/60000.0 );
```

```
        }

        CreateFuzzyId( &fidcd2id, &cd2id  );
        CreateFuzzyId( &fidcdFromDB, &cdFromDB  );

        matchpercent = FuzzyMatch( &fidcd2id, &fidcdFromDB );
        printf ("The cd's matchpercent was computed as=%f", matchpercent);
        printf ("\n");
        printf ("\n");

        printf ("Test #2 will compare two cd that are nearly the same\nexcept they have
diffent # of tracks \n");

        // put in some test values for the cd track lengths
        // since these are in ms, its basically 60000 = 1 minute
        cd2id.track[0].lengthMs = 121323;
        cd2id.track[1].lengthMs = 234565;
        cd2id.track[2].lengthMs = 566437;
        cd2id.track[3].lengthMs = 245120;
        cd2id.track[4].lengthMs = 20000;
        cd2id.track[5].lengthMs = 120386;

        cd2id.track[6].lengthMs = 323453;
        cd2id.numTracks = 7;

        for(i=1;i<cd2id.numTracks;i++) {
                printf ("CD #1: Track = %d   length in minutes = %f\n",
                        i, (float)cd2id.track[i].lengthMs/60000.0 );
        }
        printf ("\n");
        cdFromDB.track[0].lengthMs = 121323;
        cdFromDB.track[1].lengthMs = 234565;
        cdFromDB.track[2].lengthMs = 566437;
        cdFromDB.track[3].lengthMs = 245120;
        cdFromDB.track[4].lengthMs = 20000;
        cdFromDB.track[5].lengthMs = 120386;
        cdFromDB.numTracks = 6;

        for(i=1;i<cdFromDB.numTracks;i++) {
                printf ("CD #2: Track = %d   length in minutes = %f\n",
                        i, (float)cdFromDB.track[i].lengthMs/60000.0 );
        }

        CreateFuzzyId( &fidcd2id, &cd2id  );
```

```
CreateFuzzyId( &fidcdFromDB, &cdFromDB );
matchpercent = FuzzyMatch( &fidcd2id, &fidcdFromDB );

printf ("The cd's matchpercent was computed as=%f",matchpercent);
printf ("\n");
printf ("\n");
printf ("Test #3 will compare two cd that are not the same\n\n");

// put in some test values for the cd track lengths
// since these are in ms, its basically 60000 = 1 minute
cd2id.track[0].lengthMs = 34213;
cd2id.track[1].lengthMs = 334565;
cd2id.track[2].lengthMs = 231423;
cd2id.track[3].lengthMs = 134122;
cd2id.track[4].lengthMs = 2342;
cd2id.track[5].lengthMs = 3487;
cd2id.track[6].lengthMs = 9976;
cd2id.numTracks = 7;

for(i=1;i<cd2id.numTracks;i++) {
        printf ("CD #1: Track = %d   length in minutes = %f\n",
                i, (float)cd2id.track[i].lengthMs/60000.0 );
}
printf ("\n");
cdFromDB.track[0].lengthMs = 121323;
cdFromDB.track[1].lengthMs = 234565;
cdFromDB.track[2].lengthMs = 566437;
cdFromDB.track[3].lengthMs = 245120;
cdFromDB.track[4].lengthMs = 20000;
cdFromDB.track[5].lengthMs = 120386;
cdFromDB.track[6].lengthMs = 323453;
cdFromDB.numTracks = 6;


for(i=1;i<cdFromDB.numTracks;i++) {
        printf ("CD #2: Track = %d   length in minutes = %f\n",
                i, (float)cdFromDB.track[i].lengthMs/60000.0 );
}

CreateFuzzyId( &fidcd2id, &cd2id);
CreateFuzzyId( &fidcdFromDB, &cdFromDB);

matchpercent = FuzzyMatch( &fidcd2id, &fidcdFromDB );
printf ("The cd's matchpercent was computed as=%f",matchpercent);
```

}

APPENDIX B

```
/*
 * EXACT MATCH CD ID
 * - 1996 ION
 *
 *
 * by Ty Roberts
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

struct cdid{
        long    id[2];
};

typedef struct cdid cdid, *cdidPtr;

// structure of a cd track with all times stored in milliseconds

struct cdtrack{
        long    beginMs;        // start time in miliseconds
        long    endMs;                  // end time in milliseconds
        long    lengthMs;       //length in Miliseconds
};

typedef struct cdtrack cdtrack, *cdTrackPtr;

struct cd {
        short   numTracks;
        cdtrack         track[100];
};

typedef struct cd cd, *cdPtr;

void CreateUniqueId( cdidPtr cid, cdPtr cd  );

// SUBROUTINES
void CreateUniqueId( cdidPtr cid, cdPtr cd  )
{
        long            i, t, n;

        t = 0;
```

```
        n = 0;

        for(i=0;i<cd->numTracks;i++) {
                // shift left and create a MSB length thats not exact
                t += cd->track[i].lengthMs;
                n += cd->track[i].beginMs + cd->track[i].endMs;
        }
        cid->id[0] = t<<10+cd->numTracks;
        cid->id[1] = n;

}

void main(void)
{
        short i;
        short   matchtest;

        // create global structures for two complete cds with up to 100 tracks
        cd      cd2id;
        cdid cd2UID;

        cd      cdFromDB;
        cdid cdFromDBUID;

        printf ("Test #1 will compare two cd that are exactly the same\n\n");

        // put in some test values for the cd track lengths
        // since thes are in ms, its basically 60000 = 1 minute
        cd2id.track[0].beginMs = 0;
        cd2id.track[1].beginMs = 100001;
        cd2id.track[2].beginMs = 231001;
        cd2id.track[3].beginMs = 345001;
        cd2id.track[4].beginMs = 435001;
        cd2id.track[5].beginMs = 460001;
        cd2id.track[6].beginMs = 590001;

        cd2id.track[0].endMs = 100000;
        cd2id.track[1].endMs = 231000;
        cd2id.track[2].endMs = 345000;
        cd2id.track[3].endMs = 435000;
        cd2id.track[4].endMs = 460000;
        cd2id.track[5].endMs = 590000;
        cd2id.track[6].endMs = 690000;
        cd2id.track[0].lengthMs = cd2id.track[0].endMs - cd2id.track[0].beginMs;
```

```
        cd2id.track[1].lengthMs = cd2id.track[1].endMs - cd2id.track[1].beginMs;
        cd2id.track[2].lengthMs = cd2id.track[2].endMs - cd2id.track[2].beginMs;
        cd2id.track[3].lengthMs = cd2id.track[3].endMs - cd2id.track[3].beginMs;
        cd2id.track[4].lengthMs = cd2id.track[4].endMs - cd2id.track[4].beginMs;
        cd2id.track[5].lengthMs = cd2id.track[5].endMs - cd2id.track[5].beginMs;
        cd2id.track[6].lengthMs = cd2id.track[6].endMs - cd2id.track[6].beginMs;
        cd2id.numTracks = 7;

        for(i=1;i<cd2id.numTracks;i++) {
            printf ("CD #1: Track = %d   length inminutes = %f\n", i,
(float)cd2id.track[i].lengthMs/60000.0 );
        }
        printf ("\n");
        cdFromDB.track[0].beginMs = 0;
        cdFromDB.track[1].beginMs = 100001;
        cdFromDB.track[2].beginMs = 231001;
        cdFromDB.track[3].beginMs = 345001;
        cdFromDB.track[4].beginMs = 435001;
        cdFromDB.track[5].beginMs = 460001;
        cdFromDB.track[6].beginMs = 590001;
        cdFromDB.track[0].endMs = 100000;
        cdFromDB.track[1].endMs = 231000;
        cdFromDB.track[2].endMs = 345000;
        cdFromDB.track[3].endMs = 435000;
        cdFromDB.track[4].endMs = 460000;
        cdFromDB.track[5].endMs = 590000;
        cdFromDB.track[6].endMs = 690000;
        cdFromDB.track[0].lengthMs = cd2id.track[0].endMs - cd2id.track[0].beginMs;
        cdFromDB.track[1].lengthMs = cd2id.track[1].endMs - cd2id.track[1].beginMs;
        cdFromDB.track[2].lengthMs = cd2id.track[2].endMs - cd2id.track[2].beginMs;
        cdFromDB.track[3].lengthMs = cd2id.track[3].endMs - cd2id.track[3].beginMs;
        cdFromDB.track[4].lengthMs = cd2id.track[4].endMs - cd2id.track[4].beginMs;
        cdFromDB.track[5].lengthMs = cd2id.track[5].endMs - cd2id.track[5].beginMs;
        cdFromDB.track[6].lengthMs = cd2id.track[6].endMs - cd2id.track[6].beginMs;
        cdFromDB.numTracks = 7;


        for(i=1;i<cdFromDB.numTracks;i++) {
            printf ("CD #2: Track = %d   length inminutes = %f\n", i,
(float)cdFromDB.track[i].lengthMs/60000.0 );
        }

        CreateUniqueId( &cd2UID, &cd2id );
```

```
        printf( "Unique ID for CD #1 = %d%d\n", cd2UID.id[0], cd2UID.id[1] );

        CreateUniqueId( &cdFromDBUID, &cdFromDB );
        printf( "Unique ID for CD #2 = %d%d\n", cdFromDBUID.id[0],
cdFromDBUID.id[1] );

    matchtest = (cd2UID.id[0] == cdFromDBUID.id[0]) && (cd2UID.id[1] ==
cdFromDBUID.id[1]);

        printf ("The cd's match if result is non zero matchresult=%d",matchtest);

        printf ("\n");

        printf ("\n");
        printf ("Test #2 will compare two cd that are nearly the same\nexcept they have
diffent # of tracks \n");

        // put in some test values for the cd track lengths
        // since thes are in ms, its basically 60000 = 1 minute
        cd2id.track[0].beginMs = 0;
        cd2id.track[1].beginMs = 100001;
        cd2id.track[2].beginMs = 231001;
        cd2id.track[3].beginMs = 345001;
        cd2id.track[4].beginMs = 435001;
        cd2id.track[5].beginMs = 460001;
        cd2id.track[6].beginMs = 590001;
        cd2id.track[0].endMs = 100000;
        cd2id.track[1].endMs = 231000;
        cd2id.track[2].endMs = 345000;
        cd2id.track[3].endMs = 435000;
        cd2id.track[4].endMs = 460000;
        cd2id.track[5].endMs = 590000;
        cd2id.track[6].endMs = 690000;

        cd2id.track[0].lengthMs = cd2id.track[0].endMs - cd2id.track[0].beginMs;
        cd2id.track[1].lengthMs = cd2id.track[1].endMs - cd2id.track[1].beginMs;
        cd2id.track[2].lengthMs = cd2id.track[2].endMs - cd2id.track[2].beginMs;
        cd2id.track[3].lengthMs = cd2id.track[3].endMs - cd2id.track[3].beginMs;
        cd2id.track[4].lengthMs = cd2id.track[4].endMs - cd2id.track[4].beginMs;
        cd2id.track[5].lengthMs = cd2id.track[5].endMs - cd2id.track[5].beginMs;
        cd2id.track[6].lengthMs = cd2id.track[6].endMs - cd2id.track[6].beginMs;
        cd2id.numTracks = 7;

        for(i=1;i<cd2id.numTracks;i++) {
```

```
            printf ("CD #1: Track = %d   length inminutes = %f\n", i,
(float)cd2id.track[i].lengthMs/60000.0 );
        }

        printf ("\n");
        cdFromDB.track[0].beginMs = 0;
        cdFromDB.track[1].beginMs = 100001;
        cdFromDB.track[2].beginMs = 231001;
        cdFromDB.track[3].beginMs = 345001;
        cdFromDB.track[4].beginMs = 435001;
        cdFromDB.track[5].beginMs = 460001;
        cdFromDB.track[6].beginMs = 590001;

        cdFromDB.track[0].endMs = 100000;
        cdFromDB.track[1].endMs = 231000;
        cdFromDB.track[2].endMs = 345000;
        cdFromDB.track[3].endMs = 435000;
        cdFromDB.track[4].endMs = 460000;
        cdFromDB.track[5].endMs = 590000;

        cdFromDB.track[0].lengthMs = cd2id.track[0].endMs - cd2id.track[0].beginMs;
        cdFromDB.track[1].lengthMs = cd2id.track[1].endMs - cd2id.track[1].beginMs;
        cdFromDB.track[2].lengthMs = cd2id.track[2].endMs - cd2id.track[2].beginMs;
        cdFromDB.track[3].lengthMs = cd2id.track[3].endMs - cd2id.track[3].beginMs;
        cdFromDB.track[4].lengthMs = cd2id.track[4].endMs - cd2id.track[4].beginMs;
        cdFromDB.track[5].lengthMs = cd2id.track[5].endMs - cd2id.track[5].beginMs;
        cdFromDB.numTracks = 6;

        for(i=1;i<cdFromDB.numTracks;i++) {
            printf ("CD #2: Track = %d   length inminutes = %f\n", i,
(float)cdFromDB.track[i].lengthMs/60000.0 );
        }

        CreateUniqueId( &cd2UID, &cd2id  );
        printf( "Unique ID for CD #1 = %d%d\n", cd2UID.id[0], cd2UID.id[1] );

        CreateUniqueId( &cdFromDBUID, &cdFromDB  );
        printf( "Unique ID for CD #2 = %d%d\n", cdFromDBUID.id[0],
cdFromDBUID.id[1] );

        matchtest = (cd2UID.id[0] == cdFromDBUID.id[0]) && (cd2UID.id[1] ==
cdFromDBUID.id[1]);

        printf ("The cd's match if result is non zero matchresult=%d",matchtest);
```

```
        printf ("\n");
        printf ("\n");
}
```